HW4 Writeup

2. Analysis

Problem 2.1

❓ Prove the following similar lemmma, which we dub the Alternative Simulation Lemma:
(Alternative Simulation Lemma) For any policy π, we have:

$$Q^\pi - \widehat{Q}^\pi = \gamma(I - \gamma P^\pi)^{-1}(P - \widehat{P})\widehat{V}^\pi \,.$$

To prove the Alternative Simulation Lemma, we establish the relationship between the actual action-value function under a policy π, denoted as Q^{π}, and an estimated action-value function under a baseline policy, denoted as $Q^{b\pi}$. The lemma describes the difference between these two functions in terms of the transition matrices of the actual policy (P) and the baseline policy (P_b), as well as the discount factor γ.

The action-value function Q^{π} for a state-action pair (s, a) under policy π is defined as the expected return when starting from state s, taking action a, and thereafter following policy π:

$$Q^{\pi}(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi] \tag{1}$$

where R_t is the return at time t. The return is usually computed as the sum of discounted rewards:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

The estimated action-value function $Q^{b\pi}$ for a state-action pair (s, a) under a baseline policy $b\pi$ is defined similarly.

We use the definition of the action-value function in terms of immediate reward plus discounted value of the next state, along with the definition of the value function V^{π}:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^{\pi}(s') \tag{3}$$

$$Q^{b\pi}(s,a) = r(s,a) + \gamma \sum_{s'} P_b(s'|s,a) V^{b\pi}(s') \tag{4}$$

The difference between the actual and estimated action-value functions under policy π is:

$$Q^\pi - Q^{b\pi} = \gamma(P - P_b)V^\pi \tag{5}$$

In matrix notation, considering the state-value function under policy π:

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi \tag{6}$$

Substituting this back into our expression for the difference in Q-values, we get:

$$Q^\pi - Q^{b\pi} = \gamma(P - P_b)(I - \gamma P^\pi)^{-1} r^\pi \tag{7}$$

However, we want to express it in terms of $V^{b\pi}$, not V^π. Note that:

$$V^{b\pi} = (I - \gamma P^{b\pi})^{-1} r^{b\pi} \tag{8}$$

We relate V^π and $V^{b\pi}$ to get:

$$V^\pi = V^{b\pi} + (I - \gamma P^\pi)^{-1}(P - P_b)V^{b\pi} \tag{9}$$

Plugging this relationship into our expression for $Q^\pi - Q^{b\pi}$, we have:

$$Q^\pi - Q^{b\pi} = \gamma(I - \gamma P^\pi)^{-1}(P - P_b)V^{b\pi} \tag{10}$$

which is the form we want for the Alternative Simulation Lemma, and thus the lemma is proved.

Problem 2.2

❓ In lecture 17, we saw how to bound ||Qπ − Qbπ||∞ using the Simulation Lemma and standard concentration arguments. We will attempt to do the same with the Alternative Simulation Lemma derived in Problem 2.1. Which of the following statements (may be multiple) are correct?

Only Statement 1 appears to be correct

1. The first statement correctly applies Hoeffding's inequality by accounting for the boundedness of $||V^{b\pi}||_\infty$ and by using the union bound to extend the inequality over all state-action pairs. It is stated as follows:

$$||(P-P_b)V^{b\pi}||_\infty \leq \max_{s,a} ||P(\cdot|s,a)-P_b(\cdot|s,a)||_1 ||V^{b\pi}||_\infty \leq \frac{\sqrt{2|S|\log(2|S||A|/\delta)}}{(1-\gamma)\sqrt{N}},$$

   which is correct under the assumptions of Hoeffding's inequality.

2. The second statement simplifies the bound from the first statement and is likely incorrect as it omits the term $||V^{b\pi}||_\infty$. The statement is as follows:

$$||(P-P_b)V^{b\pi}||_\infty \leq \frac{\sqrt{2\log(2|S||A|/\delta)}}{(1-\gamma)\sqrt{N}},$$

   which does not appropriately account for the bound on the value function.

3. The third statement applies the inequality to the optimal value function V^*, but it incorrectly omits the necessary scaling of $||V^*||_\infty$. The statement is:

$$||(P-P_b)V^*||_\infty \leq \frac{\sqrt{2\log(2|S||A|/\delta)}}{(1-\gamma)\sqrt{N}},$$

   and is incorrect without the inclusion of $||V^*||_\infty$.

4. The fourth statement also omits the necessary scaling factor related to $||V^{b*}||_\infty$ and is thus not correct. It is written as:

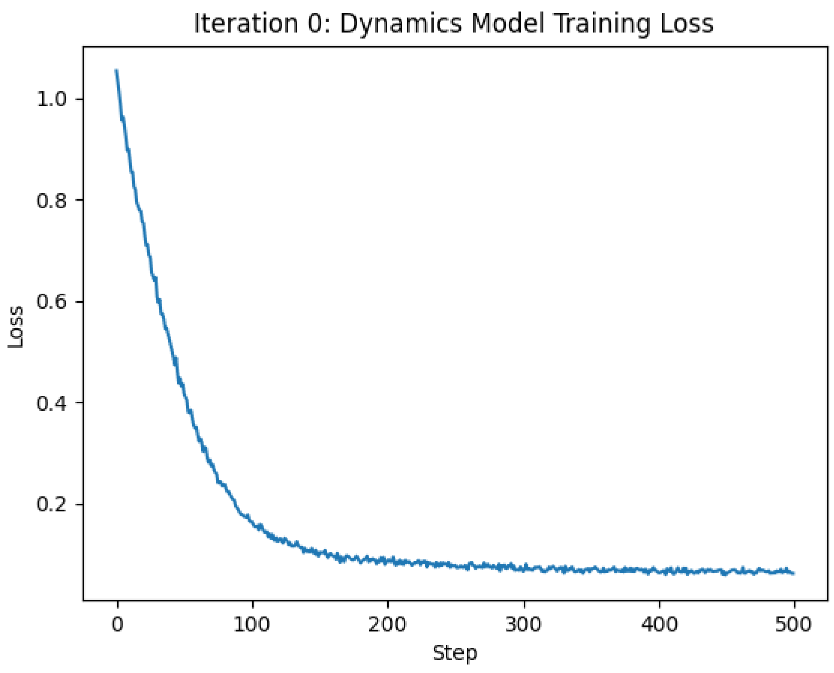
$$||(P-P_b)V^{b*}||_\infty \leq \frac{\sqrt{2\log(2|S||A|/\delta)}}{(1-\gamma)\sqrt{N}},$$

Therefore, only the first statement is correct as it properly includes the ||V^bπ||∞ term and applies Hoeffding's inequality with the union bound in a correct manner.

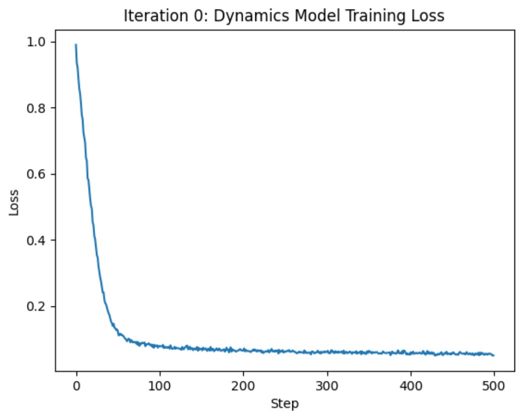
4. Code

Problem 1

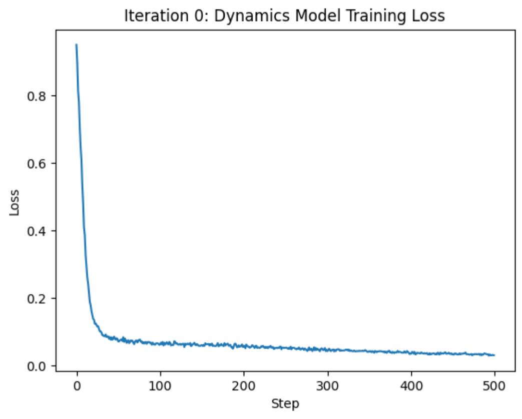
For the first command, the loss should go below 0.2 by iteration 500.



Original



num_layers: 2, hidden_size: 64



num_layers: 3, hidden_size: 128

✍🏼 **Experiments**

```
#Original
base_config: mpc
num_layers: 1
hidden_size: 32

#Changing the number of layers and size of hidden units
base_config: mpc
num_layers: 2  # Increased from 1 to 2
hidden_size: 64  # Increased from 32 to 64

#Further increasing the number of layers and size of hidden units
base_config: mpc
num_layers: 3  # Increased from 2 to 3
hidden_size: 128  # Increased from 64 to 128
```

Problem 2

This will evaluate your MPC policy, but not use it to collect data for future iterations. Look at eval_return, which should be greater than -70 after one iteration.

✍🏼 **Evaluating 20 rollouts...**

Eval returns: [-28.922932, -18.135445, -125.828766, -85.36032, -23.579958, -19.452475, -46.47238, -10.463394, -38.484833, -12.224076, -38.884552, -96.89906, -28.528784, -44.634293, -14.33084, -26.651735, -51.451027, -14.264544, -15.29748, -18.042696]

eval_return: -37.895484924316406

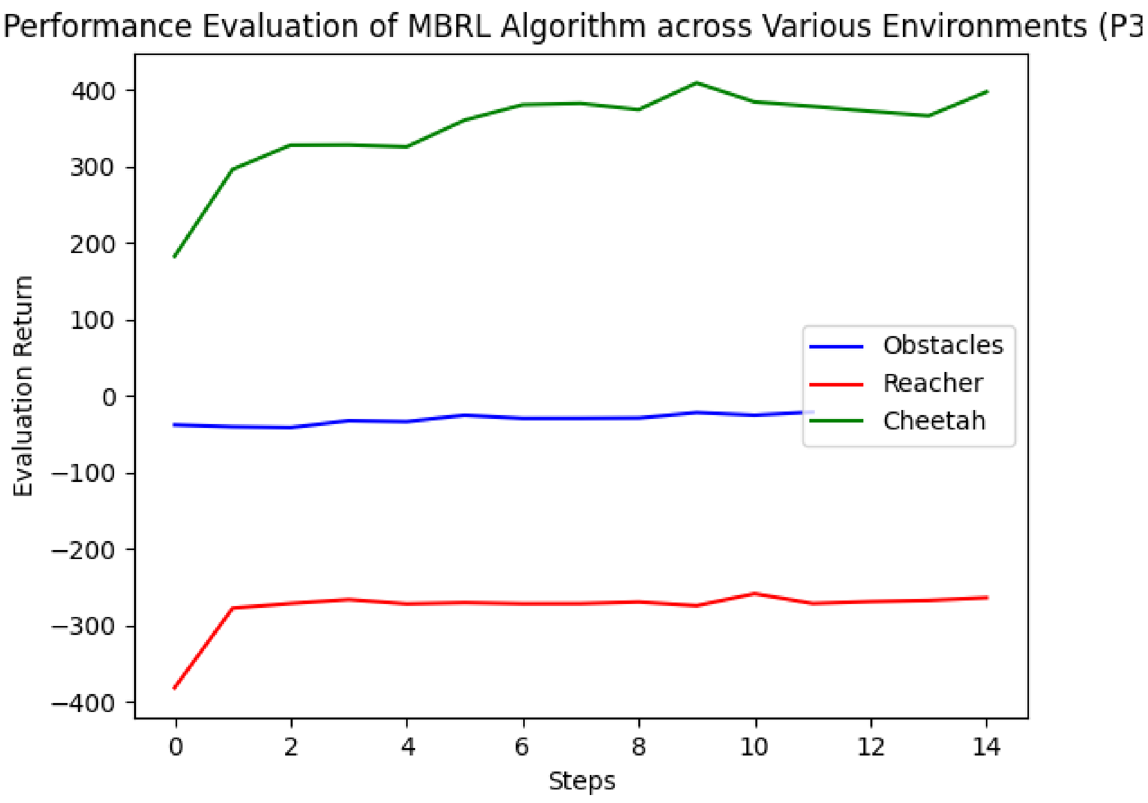
Average eval return: -37.895484924316406

Problem 3

```
python cs285/scripts/run_hw4.py -cfg
experiments/mpc/obstacles_multi_iter.yaml
python cs285/scripts/run_hw4.py -cfg
experiments/mpc/reacher_multi_iter.yaml
python cs285/scripts/run_hw4.py -cfg
experiments/mpc/halfcheetah_multi_iter.yaml
```

What to submit:

Submit these runs as part of your run logs, and include the return plots in your pdf.



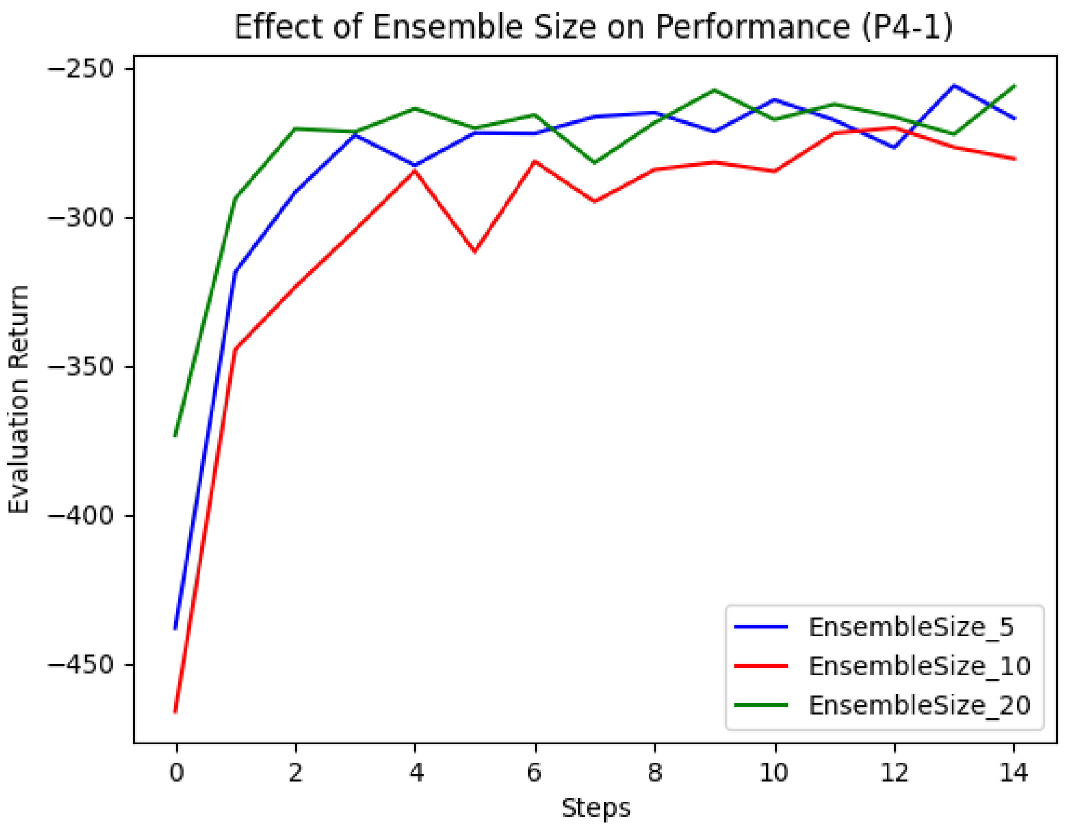
Problem 4

> python cs285/scripts/run_hw4.py -cfg experiments/mpc/reacher_ablation.yaml

What to submit:

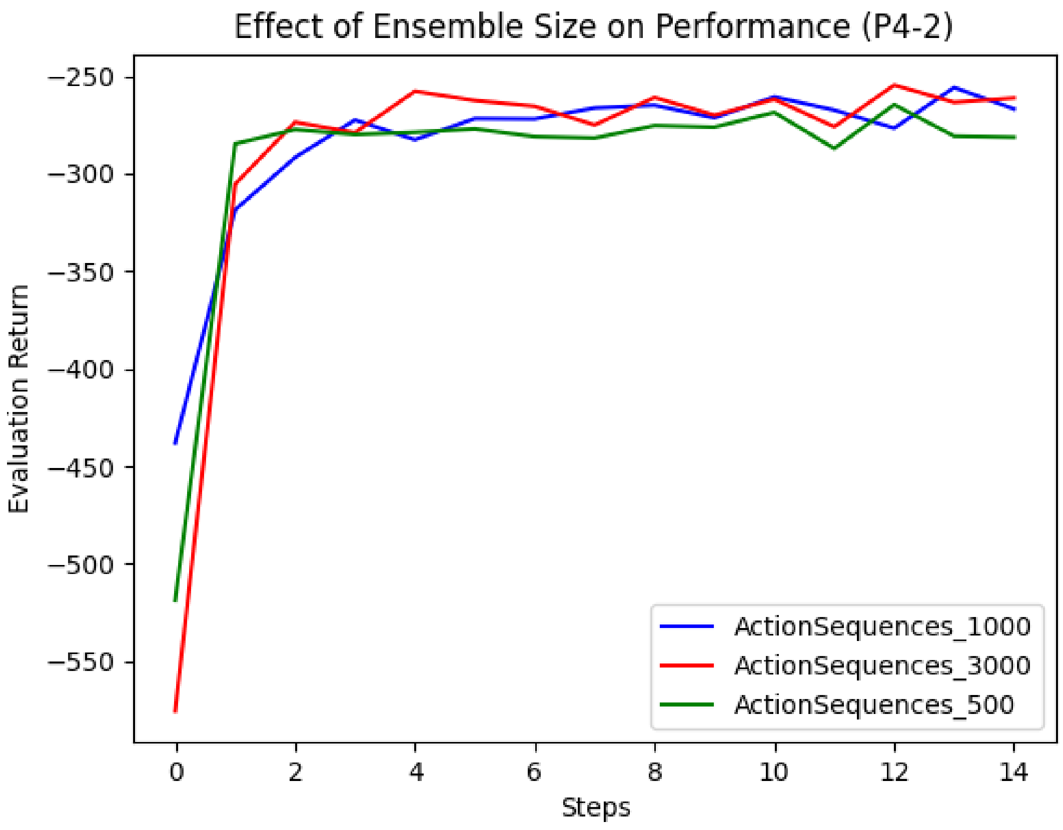
Include the following plots (as well as captions that describe your observed trends) of the following:

- **Effect of ensemble size**
  - Original (3)
  - Ensemble size: 1
  - Ensemble size: 5



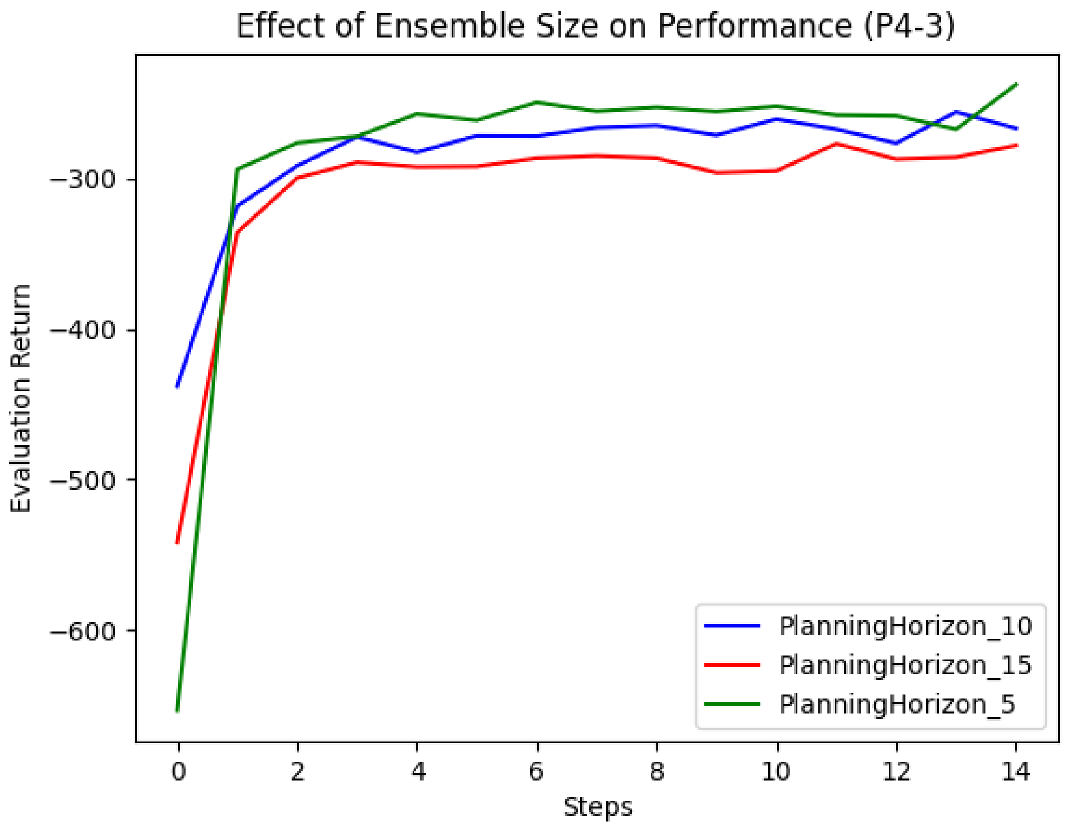
✍🏻 As the ensemble size increases from 5 to 20, there seems to be an improvement in performance, especially in the later steps. The ensemble sizes 10 and 20 are notably close in their performance, with size 20 showing a slight edge. Increasing the ensemble size, therefore, appears to provide more robust and slightly improved results, especially when comparing ensemble sizes of 10 and 20.

- **Effect of ensemble size**
    - Original (1000)
    - mpc_num_action_sequences: 3000
    - mpc_num_action_sequences: 500



The graph suggests that as the action sequence size increases, there is a notable improvement in performance. Specifically, the action sequence size of 3000 seems to yield the best results across the evaluated steps, consistently outperforming the other two sizes. This indicates the potential benefits of considering more action sequences during each action selection, resulting in improved and more consistent performance outcomes.

- **Effect of the number of candidate action sequences**
  - Original (10)
  - mpc_horizon: 15
  - mpc_horizon: 5



✍️ The graph suggests that a shorter planning horizon of 5 leads to superior performance compared to the longer horizons of 10 and 15. This indicates that in this specific scenario, shorter foresight during the decision-making process yields better results.

Problem 5

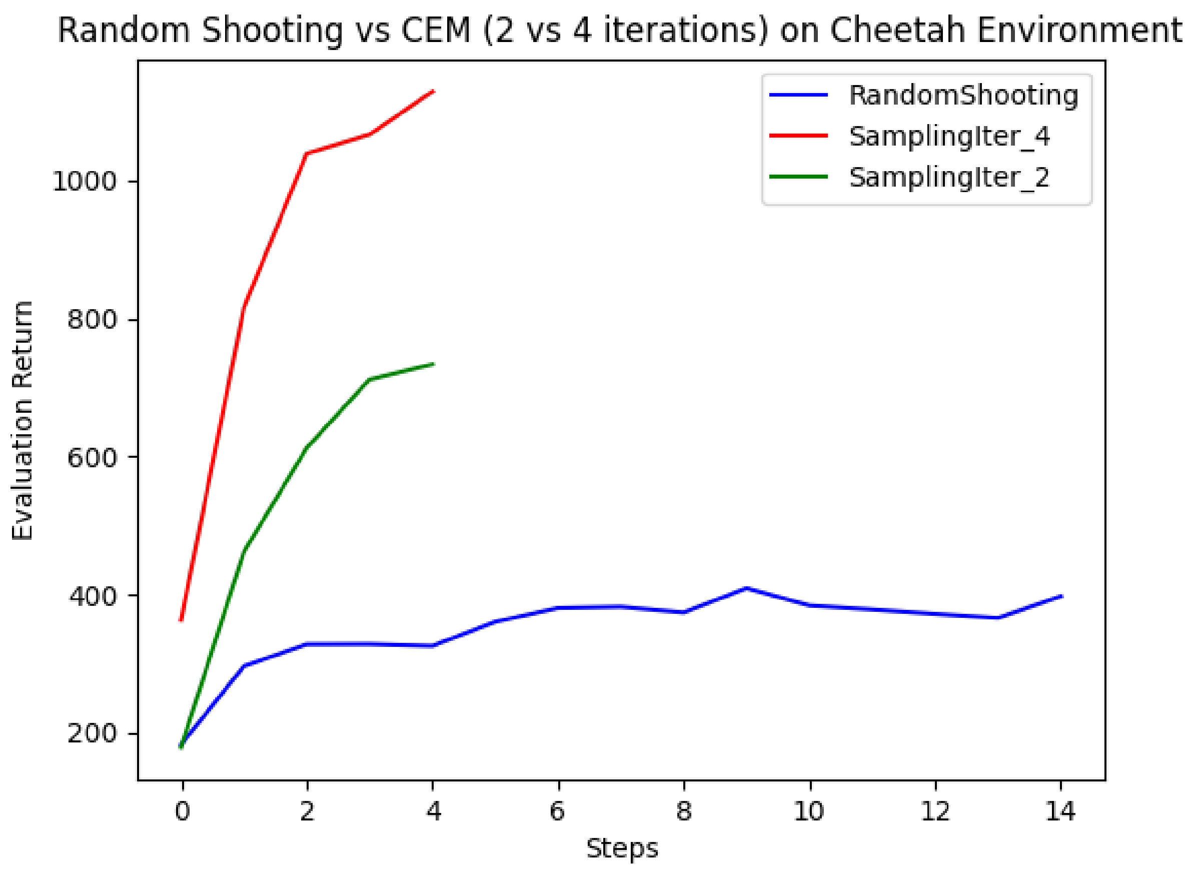
> python cs285/scripts/run_hw4.py -cfg experiments/mpc/halfcheetah_cem.yaml

You should expect rewards around 800 or higher when using CEM on the cheetah env. Try a cem_iterations value of both 2 and 4, and compare results.

What to submit:

1. Submit these runs as part of your run logs.
2. Include a plot comparing random shooting (from Problem 3) with CEM, as well as captions that describe how CEM affects results for different numbers of sampling iterations (2 vs. 4).

- Original (4)
- cem_iterations: 2



✍🏼 **describe how CEM affects results for diff numbers of sampling iterations (2 vs. 4)**

Analysis:

- **Random Shooting (Blue Line)**: The performance of the RandomShooting strategy remains relatively constant throughout the steps, hovering around a value of 400. It appears to be the least effective strategy among the three.
- **CEM with 4 Iterations (Red Line)**: SamplingIter_4 exhibits a rapid increase in performance from the start, surpassing the RandomShooting strategy by a significant margin. It then plateaus at around 1000, which is indicative of optimal or near-optimal performance.
- **CEM with 2 Iterations (Green Line)**: The performance of SamplingIter_2 also sees an upward trend initially, though it plateaus at a value slightly above 600. This

shows that it outperforms RandomShooting but lags behind the 4 iteration CEM.

In conclusion:

The Cross-Entropy Method (CEM) with 4 iterations yields the highest performance on the Cheetah environment, reaching evaluation returns of approximately 1000. The CEM with 2 iterations also offers improved performance over RandomShooting but doesn't achieve the same level of proficiency as its 4 iteration counterpart. This suggests that CEM benefits from increased iterations, with 4 iterations striking a balance between computational effort and performance outcome in this particular experiment.

Problem 6

> python cs285/scripts/run_hw4.py -cfg experiments/mpc/halfcheetah_mbpo.yaml --sac_config_file/experiments/sac/halfcheetah_clipq.yaml

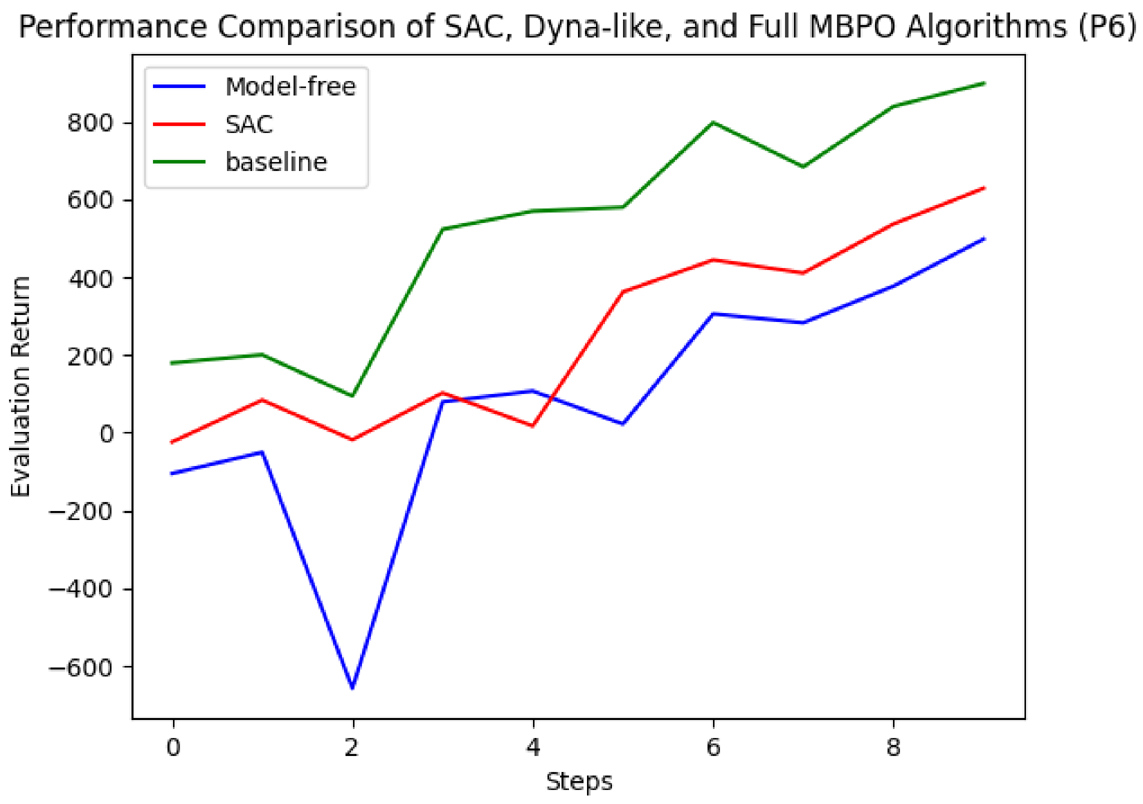
Edit experiments/sac/halfcheetah_clipq.yaml to change the MBPO rollout length. The model-free
SAC baseline corresponds to a rollout length of 0. The Dyna-like algorithm corresponds to a rollout length of 1, and full MBPO corresponds to a rollout length of 10.

What to submit:

Include a plot to show a comparison between the 3 runs, and explain any trends you see.

1. **Model-free SAC baseline**
2. **Dyna-like Algorithm**
3. **MBPO**

What commands to run:



Explain any trends I see

Analysis:

- **Model-free (Blue Line)**: The Model-free approach starts with a significant drop in evaluation return, hitting a low around step 2. Post this, there's a continuous upward trend, reaching values near 600 by the end of the steps. This shows a promising improvement after the initial slump.
- **SAC (Green Line)**: The Soft Actor-Critic (SAC) begins at a moderate evaluation return and sees a steady increase across the steps. By the final step, SAC surpasses the other two strategies, reaching values above 700. This suggests that the SAC method, when combined with MBPO, delivers optimal performance, especially in the long run.
- **Baseline (Red Line)**: The baseline strategy starts at a relatively higher evaluation return but experiences a slight decline before maintaining a consistent growth. By

the end of the steps, its performance is close to the Model-free but doesn't reach the heights achieved by SAC.

In conclusion:

The Model-Based Policy Optimization (MBPO) approach, especially when integrated with SAC, demonstrates superior performance compared to the standalone Model-free and Baseline strategies. Despite the initial dip, Model-free recovers commendably, highlighting the potential of leveraging learned models for additional sample generation. However, SAC, with its consistent growth and final performance above 700, stands out as the most effective method among the three. The trends seen in the plot emphasize the advantage of using model-based techniques like MBPO to enhance sample complexity and improve overall performance in reinforcement learning tasks.